SYSTEM AND METHOD FOR DYNAMIC DEVICE DRIVER SUPPORT IN AN OPEN SOURCE OPERATING SYSTEM

Inventors: Chieng-Hwa Lin
10729 Chestnut Ridge Road
Austin, Texas 78726

Sanjay Rao
1773 Wells Branch Parkway #1308
Austin, Texas 78728

Assignee: DELL PRODUCTS, L.P.

BAKER BOTTS L.L.P.
One Shell Plaza
910 Louisiana
Houston, Texas 77002-4995

Attorney's Docket: 016295.0732
DC-3207

SYSTEM AND METHOD FOR DYNAMIC DEVICE DRIVER SUPPORT IN AN OPEN SOURCE OPERATING SYSTEM

TECHNICAL FIELD

The present disclosure relates generally to the field of computer systems and, more specifically, to a system and method for dynamic device driver support for the kernel of an open source operating system.

BACKGROUND

The operating system of a computer server is an interface between the hardware and the software applications of the computer system. Each operating system, sometimes referred to as a kernel, typically includes an element of software code known as a device driver. The term kernel is sometimes used to refer to the lowest level of the operating system. The device driver of the kernel typically resides in the lowest level of the operating system and provides the low level interface between the hardware elements of the computer system and the operating system to direct the operation of the hardware elements of the computer system.

Because the device driver of an operating system provides an interface between the hardware of the computer system and the application programs of the computer systems, the application programs of the computer system are able to interface with the hardware without the necessity of making functions calls that are specific to the hardware architecture of each model of computer system. Through function calls to the operating system or kernel, application programs may request certain action by the hardware resources of the computer system, including such hardware resources as the CPU, storage devices, memory, and input/output devices. Once such a function call is received by the operating system, the device driver of the operating system or kernel interfaces with the hardware resources of the computer system. A device driver, which serves as a software interface between the application and the hardware architecture of the computer system, may be written specifically for the hardware architecture of the

computer system. The operating system communicates with the device driver of the computer system to request certain functions of the hardware of the computer system, including CPU scheduling functions, interrupt functions, memory management functions, memory storage and retrieval functions, and device input and output functions.

Open source operating systems, such as Linux, a standardized version of which is marketed by Red Hat, Inc. of Durham, North Carolina, have come into more common use in computer systems. Open source software is characterized by freely available source code. In the case of an open source operating system, the source code of the operating system is accessible to users of the operating system. Because the source code of an open source operating system is available, the software may be readily modified to accommodate the needs or desires of the user. Even the most sensitive portions of an open source operating system, including the kernel, may be modified. After a modification by the user, the user may choose to distribute the source code to others, leading to a gradual evolution in the product. It is theorized that the ability to modify and distribute open source software leads to the faster correction of bugs and improvement of the functionality of the software, as compared with proprietary software packages. The source code for the Red Hat Linux operating system may be downloaded from the Internet or the software may be purchased from a vendor in a packaged format.

With respect to the Linux open source operating system, the Linux operating system runs on a number of hardware platforms, including computer systems platforms for desktop computer systems and server systems. A number of vendors package and sell a standardized version of the Linux operating system. This packaged system includes application programs, installation tools, utility programs, development tools, graphical interface, and the device drivers. Even though a standardized version of an open source operating system may be purchased from a third party vendor, the operating system, including the kernel , may be readily modified by the user to create a customized operating system.

Computer system manufacturers are now offering the option of installing open source operating systems in computers systems, including server systems. As part of the open source operating

system, the computer system manufacturer will typically include in the operating system package a device driver that is specific to the computer system model or family of computer system models of the computer system manufacturer. One difficulty of providing an open source driver as part of an open source operating system is that the source code of the open source driver may contain proprietary information concerning the hardware architecture of the computer system or the family of computer systems of the computer system manufacturer. A possible solution to this problem is to provide to users a precompiled device driver. The user or a competitor would not be able to identify in the compiled device driver the proprietary information that could possibly be found in the source code of the device driver.

The option of providing a precompiled device driver is not completely satisfactory. In the Linux operating system, for example, the precompiled device driver only works with the kernel that the driver is compiled against. When the device driver is compiled for the sake of execution, the device driver will expect to receive function calls from the kernel that are named according to a naming convention in which the suffix of the function call name is specific to the Linux kernel. A change to the source code of the kernel, which would necessitate the recompilation of the kernel, results in a change to the names of the function calls of the kernel. In this instance, the names of the function calls of the kernel would not match the names expected by the compiled device driver. As a result, a device driver only works for the kernel that the device driver is compiled against because the function call names must match between the compiled kernel and the compiled device driver.

Although a hardware vendor can supply precompiled device drivers to support a few popular Linux kernels, many other kernels will be left unsupported. If a computer manufacturer, wants its device driver to support all Linux kernels, source code for the device driver must be provided to vendors of standardized version of the open source operating system so the source code can be distributed to customers, enabling the customers to compile the device driver against the kernel of the customer's operating system. The difficulty with the approach of distributing an open source driver is that if proprietary

information concerning the products of the hardware manufacturer is included in the open source driver, this information may become known to others.

SUMMARY

The present disclosure concerns a system and method for providing device driver support in an open source operating system. A device driver for a computer system that includes an open source operating system, including an open source kernel, is constructed from an open source service layer and a set of precompiled driver modules. Once compiled against the kernel of the operating system, the service layer provides an interface between the kernel of the operating system and the applicable driver modules. The set of driver modules includes a number of individual driver modules, with each of the individual driver modules being specific to a hardware architecture of the computer system.

The kernel of the operating system and the compiled service layer are able to pass between one another function calls that are named according to a naming convention. The function calls passed between the kernel of the operating system and the compiled service layer are not specific to the hardware architecture of the computer system. The compiled service layer is able to pass to the appropriate driver module commands that are specific to the hardware architecture of the computer system. The service layer thus provides an interface between the open source operating system and a set of precompiled device drivers. If the kernel of the operating system is modified, the service layer is recompiled against the modified kernel.

A technical advantage of the present disclosure is the provision of a device driver for an open source operating system that permits the computer system manufacturer to provide device drivers for its computer systems, while preventing the disclosure of sensitive proprietary information in those device drivers. The device driver of the present invention includes a portion that is provided in an open source format. This portion can be compiled against the kernel of the operating system to satisfy the naming convention of the kernel function calls of the open source operating system. Another portion of the device driver includes precompiled driver modules. The precompiled driver modules may contain proprietary information concerning the hardware architecture. The communication between the kernel and the compiled service layer is not specific to the hardware architecture of the computer system, while the communication

between the compiled service layer and the compiled driver modules may be specific to the hardware architecture of the computer system.

Another technical advantage of the present invention is a method for distributing device driver software to users of a computer system having an open source operating system. A device driver having an open source service layer is provided. The provision of the open source service layer permits a user to modify or replace the kernel of the operating system. Once the user modifies the kernel of the operating system, the user can rebuild the device driver of the computer system by compiling the service layer against the modified operating system kernel. In this manner, the user can continue to modify the open source operating system, while proprietary information of the computer system manufacturer is protected against disclosure and resides in the precompiled driver modules of the device driver.

Other technical advantages will be apparent to those of ordinary skill in the art in view of the following specification, claims, and drawings.

BRIEF DESCRIPTION OF THE DRAWINGS

Figure 1 is a block diagram of the logical relationship of software elements of the present disclosure; and

Figure 2 is a flow diagram of the method for loading a device driver in a computer system having an open source operating system.

DETAILED DESCRIPTION

The present disclosure concerns a method and system for dynamic device driver support to accommodate modifications to the open source kernel of an open source operating system. An open source operating system will include a device driver that serves as an interface between the kernel of the operating system and the hardware of the computer system. The device driver of the present invention includes two elements, the first of which is a set of one or more driver modules. The driver modules are compiled, executable code. The second element of the device driver is an open source service layer. The open source service layer is an intermediate software layer that serves as a software interface between the kernel of the operating system and the driver modules of the device driver.

When the open source service layer is compiled against the kernel of the operating system, the compiled service layer is configured with respect to the kernel. As such, the compiled service layer recognizes the naming convention of the function calls from the kernel. The compiled service layer interacts with the compiled driver modules and serves as an intermediary between the driver modules, which are specific to the hardware architecture of the computer system, and the kernel. Because of the presence of the compiled service layer between the kernel and the compiled driver modules, the compiled driver modules do not have to recognize changes to the kernel of the operating system. The service layer can be recompiled, if necessary, against the kernel. As such, the function of the driver modules is not dependent on the naming convention of the function calls from kernel, and modifications to the kernel do not affect the operation of the driver modules.

Shown in Figure 1 is a block diagram of the logical relationship of the device driver modules, operating system service layer, and operating system kernel. A device driver 12 includes an operating system service layer 12 and one or more device driver modules 16 – 22. Each device driver module is associated with the hardware architecture of a computer system model of a family of computer systems. As an example, a computer manufacturer may have thirty or more computer system models within a family of server systems. Each computer system model will have an associated device driver module.

Each of the device driver modules is provided to users in executable, or compiled, format. Because the device driver modules are in an executable format, it is difficult, if not impossible, to identify the proprietary content within the module concerning the hardware architecture of the computer system or family of computer systems.

Also shown in Figure 1 is an operating system service layer 14. The operating system service layer is provided to users in an open source format. As shown in Figure 1, the operating system service layer 14 serves as an interface between operating system kernel 10 and device driver modules 16 – 22. Functions calls to device driver 12 are received by operating system service layer 14, which processes the function calls and interfaces with proper device driver module to complete the requested function call directed to the hardware of the computer system. In the case of a device driver that includes several driver modules, operating system service layer 14 will interact with only the device driver module associated with the hardware architecture of the computer system model.

Shown in Figure 2 is a series of method steps for establishing a device driver in a computer system that includes an open source operating system. At step 30, an attempt is made to load the existing device driver of the operating system. The computer manufacturer will typically include with the computer system a number of pre-compiled device drivers that are associated with standardized versions of the open source operating system. As an example, the computer manufacturer may choose to include as part of an open-source Linux operating system, pre-compiled device drivers that are associated with popular, standardized versions of the Linux operating system, including, for example, Linux 7.0 and Linux 7.1. As part of the loading operation of step 30, the computer system determines if a pre-compiled driver is associated with the kernel of the operating system. If a pre-compiled device driver exists that is associated with the kernel of the operating system, the device driver is loaded.

At step 32, it is determined whether the load operation was successful. The load operation is successful if a pre-compiled device driver exists that is associated with the kernel of the operating system. As such, the load operation is successful if the operating system is a standardized open source operating

system and the computer manufacturer has provided to the user a pre-compiled device driver that is associated with the kernel of the standardized open source operating system. If it is determined that the load operation is successful at step 32, the process of establishing a device driver in the computer system concludes at step 34. If it is determined at step 32 that the load operation of step 30 is unsuccessful, the kernel of the open source operating system is not associated with or supported by any of the existing pre-compiled device drivers of the computer system. The open source kernel may not be recognized or associated with any of the pre-compiled device drivers of the computer system because the open source kernel may be a newly standardized kernel that is not yet supported by a pre-compiled device driver or because a user has made a user-specific modification to the device. In either event, a dynamic device driver, which can be associated with the unrecognized or modified kernel, must be built.

At step 36, the open source service layer, which is shown as element 14 in Figure 1, is compiled against the kernel of the operating system. If the compilation is determined to be unsuccessful at step 38, processing ends and the user is notified at step 40 that the kernel of the operating system cannot be supported by the computer system. If the compilation is successful, a link operation is performed at step 42 in which the compiled service layer is linked to the compiled driver modules, which are shown in Figure 1 as compiled driver modules 106 – 112, forming a dynamic device driver. If it is determined at step 44 that the link operation is unsuccessful, processing ends and the user is notified at step 40 that the kernel of the operating system cannot be supported by the computer system. Following a successful link operation, the linked compiled service layer and the compiled driver modules comprise the dynamic device driver. The dynamic device driver is next loaded at step 46. If the load of the dynamic device driver is determined at step 48 to be unsuccessful, processing ends and the user is notified at step 40 that the kernel of the operating system cannot be supported by the computer system. If the load of the dynamic device driver is successful, processing ends. The successful load of the dynamic device driver permits the kernel of the operating system to function with and make calls to the hardware of the computer system.

The invention described herein is applicable to any information handling system that includes an operating system and a hardware system, including any computer system, server, router, or any other instrumentality or aggregate of instrumentalities that is operable to compute, classify, process, transmit, receive, retrieve, originate, switch, store, display, manifest, detect, record, reproduce, handle, or utilize any form of information, intelligence, or data for business, scientific, control or other purposes.

Although the present disclosure has been described in detail, it should be understood that various changes, substitutions, and alterations can be made hereto without departing from the spirit and the scope of the invention as defined by the appended claims.